Java Test

1. Given

```
public class Dec26 {
public static void main(String[] args) {
short a1 = 6;
new Dec26().go(a1);
new Dec26().go(new Integer(7));
}
void go(Short x) { System.out.print("S "); }
void go(Long x) { System.out.print("L "); }
void go(int x) { System.out.print("i "); }
void go(Number n) { System.out.print("N "); }
}
```

What is the result?
A. i L
B. i N
C. S L
D. S N
E. Compilation fails.
F. An exception is thrown at runtime.

2. Given:

```
1. public class Fellowship {
2. public static void main(String[] args) {
3. // insert code here
4. }
5. }
6. class Numinor {
7. enum Members {
8. HOBBITS(48), ELVES(74), DWARVES(50);
9. int height;
10. Members(int h) { height = h; }
11. int getHeight() { return height; }
12. }
13. }
```

And these four lines of code to be inserted, independently at line 3:

```
I. int h0 = Numinor.Members.HOBBITS.getHeight();
II. int h1 = Numinor.Members.getHeight();
III. int h2 = Members.HOBBITS.getHeight();
IV. int h3 = Members.height;
```

Which are true? (Choose all that apply.)
A. Line I will compile.
B. Line II will compile.
C. Line III will compile.

D. Line IV will compile.
E. Class Numinor will NOT compile.

3. Given:

```
2. public class Volume {
3. Volume v;
4. int size;
5. public static void main(String[] args) {
6. Volume myV = new Volume();
7. final Volume v2;
8. v2 = myV.doStuff(myV);
9. v2.v.size = 7;
10. System.out.print(v2.size);
11. }
12. Volume doStuff(Volume v3) {
13. v3.size = 5;
14. v3.v = new Volume();
15. return v3;
16. } }
```

What is the result? (Choose all that apply.)

A. 5
B. 7
C. Compilation fails due to an error on line 8.
D. Compilation fails due to an error on line 9.
E. Compilation fails due to an error on line 13.
F. Compilation fails due to an error on line 14.

4. Given:

```
3. public class BirdHouse {
4. public static void main(String[] args) {
5. String r = "0";
6. int x = -1, y = -5;
7. if(x < 5)
8. if(y > 0)
9. if(x > y)
10. r += "1";
11. else r += "2";
12. else r += "3";
13. else r += "4";
14. System.out.println(r);
15. } }
```

What is the result?

A. 0
B. 01
C. 02
D. 03

E. 013
F. 023
G. Compilation fails.

5. Given:

```
1. class c1 { }
2. class c2 { }
3. interface i1 { }
4. interface i2 { }
5. class A extends c2 implements i1 { }
6. class B implements i1 implements i2 { }
7. class C implements c1 { }
8. class D extends c1, implements i2 { }
9. class E extends i1, i2 { }
10. class F implements i1, i2 { }
```

What is the result? (Choose all that apply.)
A. Class A does not compile.
B. Class B does not compile.
C. Class C does not compile.
D. Class D does not compile.
E. Class E does not compile.
F. Class F does not compile.
G. Compilation succeeds for all of the classes.

6. Given:

```
2. class SuperCool {
3. static String os = "";
4. void doStuff() { os += "super "; }
5. }
6. public class Cool extends SuperCool {
7. public static void main(String[] args) {
8. new Cool().go();
9. }
10. void go() {
11. SuperCool s = new Cool();
12. Cool c = (Cool)s;
13. // insert code here
14. }
15. void doStuff() { os += "cool "; }
16. }
```

If the rest of the code compiles, which line(s) of code, inserted independently at line 13, compile? (Choose all that apply.)
A. c.doStuff();
B. s.doStuff();
C. this.doStuff();
D. super.doStuff();

E. c.super.doStuff();
F. s.super.doStuff();
G. this.super.doStuff();
H. There are other errors in the code.

7. Given:

```
 5. static String s = "";
 6. public static void main(String[] args) {
 7. try { doStuff(args); }
 8. catch (Error e) { s += "e "; }
 9. s += "x ";
10. System.out.println(s);
11. }
12. static void doStuff(String[] args) {
13. if(args.length == 0)
14. throw new IllegalArgumentException();
15. s += "d ";
16. }
```

And, if the code compiles, and given a java invocation with no arguments, what is the result? (Choose all that apply.)
A. d x
B. e x
C. d e x
D. Compilation fails due to an error on line 8.
E. Compilation fails due to an error on line 12.
F. Compilation fails due to an error on line 14.
G. An uncaught IllegalArgumentException is thrown

8. Which are true? (Choose all that apply.)

A. For a specific object, it's NOT possible for finalize() to be invoked more than once.
B. It's possible for objects, on whom finalize() has been invoked by the JVM, to avoid the GC.
C. Overriding finalize() ensures that objects of that type will always be GCed when they become eligible.
D. The finalize() method is invoked only for GC-eligible objects that are NOT part of "islands of isolation."
E. For every object that the GC considers collecting, the GC remembers whether finalize() has been invoked for that specific object.

9. Given that:
Exception is the superclass of IOException and
IOException is the superclass of FileNotFoundException and

```
 2. import java.io.*;
 3. class Author {
 4. protected void write() throws IOException { }
 5. }
 6. public class Salinger extends Author {
```

```
 7. private void write(int x) { }
 8. protected void write(long x) throws FileNotFoundException { }
 9. protected void write(boolean x) throws Exception { }
10. protected int write(short x) { return 7; }
11. public void write() { }
12. }
```

What is the result? (Choose all that apply.)

A. Compilation succeeds.
B. Compilation fails due to an error on line 7.
C. Compilation fails due to an error on line 8.
D. Compilation fails due to an error on line 9.
E. Compilation fails due to an error on line 10.
F. Compilation fails due to an error on line 11.

10. Given:

```
 2. class Chilis {
 3. Chilis(String c, int h) { color = c; hotness = h; }
 4. String color;
 5. int hotness;
 6. public boolean equals(Object o) {
 7. if(this == (Chilis)o) return true;
 8. return false;
 9. }
10. public String toString() { return color + " " + hotness; }
11. }
```

If instances of class Chilis are to be used as keys in a Map, which are true? (Choose all that apply.)

A. Without overriding hashCode(), the code will not compile.
B. As it stands, the equals() method has been legally overridden.
C. It's possible for such keys to find the correct entries in the Map.
D. It's NOT possible for such keys to find the correct entries in the Map.
E. As it stands, the Chilis class legally supports the equals() and hashCode() contracts.
F. If hashCode() was correctly overridden, it would make retrieving Map entries by key easier.

11. Given:

```
 2. public class Contact {
 3. private String name;
 4. private String city;
 5. String getName() { return name; }
```

Practice Exam 1 65

```
 6. void setName(String n) { name = n; }
 7. void setCity(String c) {
 8. if(c == null) throw new NullPointerException();
 9. city = c;
10. }
11. String getCity() { return city; }
```

12. }

Which are true? (Choose all that apply.)

A. Compilation fails.
B. The class is well encapsulated.
C. The setCity() method is an example of loose coupling.
D. The setCity() method has better encapsulation than setName().
E. The setCity() method is cohesive; the setName() method is not.

12. Given:

```
1. interface Syrupable {
2. void getSugary();
3. }
4. abstract class Pancake implements Syrupable { }
5.
6. class BlueBerryPancake implements Pancake {
7. public void getSugary() { ; }
8. }
9. class SourdoughBlueBerryPancake extends BlueBerryPancake {
10. void getSugary(int s) { ; }
11. }
```

Which are true? (Choose all that apply.)

A. Compilation succeeds.
B. Compilation fails due to an error on line 2.
C. Compilation fails due to an error on line 4.
D. Compilation fails due to an error on line 6.
E. Compilation fails due to an error on line 7.
F. Compilation fails due to an error on line 9.
G. Compilation fails due to an error on line 10.

13. Given:

```
1. public class Endless {
2. public static void main(String[] args) {
3. int i = 0;
4. short s = 0;
5. for(int j = 0, k = 0; j < 3; j++) ;
6. for(int j = 0; j < 3; counter(j)) ;
7. for(int j = 0, int k = 0; j < 3; j++) ;
8. for(; i < 5; counter(5), i++) ;
9. for(i = 0; i < 3; i++, System.out.print("howdy ")) ;
10. }
11. static int counter(int y) { return y + 1; }
12. }
```

What is the result? (Choose all that apply.)

A. howdy howdy howdy
B. The code runs in an endless loop.
C. Compilation fails due to an error on line 5.

D. Compilation fails due to an error on line 6.
E. Compilation fails due to an error on line 7.
F. Compilation fails due to an error on line 8.
G. Compilation fails due to an error on line 9.

14. Given:

```
 2. class Big {
 3. void doStuff(int x) { }
 4. }
 5. class Heavy extends Big {
 6. // void doStuff(byte b) { }
 7. // protected void doStuff(int x) throws Exception { }
 8. }
 9. public class Weighty extends Heavy {
10. // void doStuff(int x) { }
11. // String doStuff(int x) { return "hi"; }
12. // public int doStuff(int x) { return 7; }
13. // private int doStuff(char c) throws Error { return 1; }
14. }
```

Which method(s), if uncommented independently, compile? (Choose all that apply.)
A. Line 6
B. Line 7
C. Line 10
D. Line 11
E. Line 12
F. Line 13

15. Given:

```
 1. public class Grids {
 2. public static void main(String[] args) {
 3. int [][] ia2;
 4. int [] ia1 = {1,2,3};
 5. Object o = ia1;
 6. ia2 = new int[3][3];
 7. ia2[0] = (int[])o;
 8. ia2[0][0] = (int[])o;
 9. } }
```

What is the result? (Choose all that apply.)
A. Compilation fails due to an error on line 4.
B. Compilation fails due to an error on line 5.
C. Compilation fails due to an error on line 6.
D. Compilation fails due to an error on line 7.
E. Compilation fails due to an error on line 8.
F. Compilation succeeds and the code runs without exception.
G. Compilation succeeds and an exception is thrown at runtime.

16. Given:

```
 3. public class OffRamp {
```

```
4. public static void main(String[] args) {
5. int [] exits = {0,0,0,0,0,0};
6. int x1 = 0;
7.
8. for(int x = 0; x < 4; x++) exits[0] = x;
9. for(int x = 0; x < 4; ++x) exits[1] = x;
10.
11. x1 = 0; while(x1++ < 3) exits[2] = x1;
12. x1 = 0; while(++x1 < 3) exits[3] = x1;
13.
14. x1 = 0; do { exits[4] = x1; } while(x1++ < 7);
15. x1 = 0; do { exits[5] = x1; } while(++x1 < 7);
16.
17. for(int x: exits)
18. System.out.print(x + " ");
19. } }
```

What is the result?

A. 3 3 2 2 6 6
B. 3 3 3 2 7 6
C. 3 3 3 2 7 7
D. 4 3 3 2 7 6
E. 4 3 3 2 7 7
F. Compilation fails.

17. Given:

```
2. import java.util.*;
3. public class HR {
4. public static void main(String[] args) {
5. List<Integer> i = new Vector<Integer>();
6. i.add(3); i.add(2); i.add(5);
7. int ref = 1;
8. doStuff(ref);
9. System.out.println(i.get(ref));
10. }
11. static int doStuff(int x) {
12. return ++x;
13. } }
```

What is the result?

A. 2
B. 3
C. 5
D. Compilation fails.
E. An exception is thrown at runtime.

18. Given:

```
2. import java.util.*;
3. public class Vinegar {
4. public static void main(String[] args) {
```

```
 5. Set<Integer> mySet = new HashSet<Integer>();
 6. do1(mySet, "0"); do1(mySet, "a");
 7. do2(mySet, "0"); do2(mySet, "a");
 8. }
 9. public static void do1(Set s, String st) {
10. s.add(st);
11. s.add(Integer.parseInt(st));
12. }
13. public static void do2(Set<Integer> s, String st) {
14. s.add(st);
15. s.add(Integer.parseInt(st));
16. } }
```

Which are true? (Choose all that apply.)

A. Compilation succeeds.
B. Compilation fails due to an error on line 6.
C. Compilation fails due to an error on line 13.
D. Compilation fails due to an error on line 14.
E. Compilation fails due to an error on line 15.
F. If only the line(s) of code that don't compile are removed, the code will run without exception.
G. If only the line(s) of code that don't compile are removed, the code will throw an exception.

19. Given:

```
 3. class Employee {
 4. private String name;
 5. void setName(String n) { name = n; }
 6. String getName() { return name; }
 7. }
 8. interface Mungeable {
 9. void doMunging();
10. }
11. public class MyApp implements Mungeable {
12. public void doMunging() { ; }
13. public static void main(String[] args) {
14. Employee e = new Employee();
15. e.setName("bob");
16. System.out.print(e.getName());
17. } }
```

Which are true? (Choose all that apply.)

A. MyApp is-a Employee.
B. MyApp is-a Mungeable.
C. MyApp has-a Employee.
D. MyApp has-a Mungeable.
E. The code is loosely coupled.
F. The Employee class is well encapsulated.

20. Given that FileNotFoundException extends IOException, and given:

```
2. import java.io.*;
3. public class MacPro extends Laptop {
4. public static void main(String[] args) {
5. new MacPro().crunch();
6. }
7. // insert code here
8. }
9. class Laptop {
10. void crunch() throws IOException { }
11. }
```

Which method(s), inserted independently at line 7, compile? (Choose all that apply.)

A. void crunch() { }
B. void crunch() throws Exception { }
C. void crunch(int x) throws Exception { }
D. void crunch() throws RuntimeException { }
E. void crunch() throws FileNotFoundException { }

21. Given:

```
2. class Horse {
3. String hands = "15";
4. }
5. class GaitedPony extends Horse {
6. static String hands = "14";
7. public static void main(String[] args) {
8. String hands = "13.2";
9. String result = new GaitedPony().getSize(hands);
10. System.out.println(" " + result);
11. }
12. String getSize(String s) {
13. System.out.print("hands: " + s);
14. return hands;
15. } }
```

What is the result?

A. 14
B. 15
C. hands: 13.2 14
D. hands: 13.2 15
E. Compilation fails.
F. An exception is thrown at runtime

22. Given:

```
2. public class Humping {
3. public static void main(String[] args) {
4. String r = "-";
5. char[] c = {'a', 'b', 'c', 'z'};
6. for(char c1: c)
7. switch (c1) {
```

```
 8. case 'a': r += "a";
 9. case 'b': r += "b"; break;
10. default: r += "X";
11. case 'z': r+= "z";
12. }
13. System.out.println(r);
14. } }
```

What is the result?

A. -abXz
B. -abbXz
C. -abbXzz
D. -abbXzXz
E. Compilation fails due to a single error.
F. Compilation fails due to multiple errors.

23. Given:

```
 1. import java.util.*;
 2. public class Garage {
 3. public static void main(String[] args) {
 4. Map<String, String> hm = new HashMap<String, String>();
 5. String[] k = {null, "2", "3", null, "5"};
 6. String[] v = {"a", "b", "c", "d", "e"};
 7.
 8. for(int i=0; i<5; i++) {
 9. hm.put(k[i], v[i]);
10. System.out.print(hm.get(k[i]) + " ");
11. }
12. System.out.print(hm.size() + " " + hm.values() + "\n");
13. } }
```

What result is most likely?

A. a b c a e 4 [c, b, a, e]
B. a b c d e 4 [c, b, a, e]
C. a b c d e 4 [c, d, b, e]
D. a b c, followed by an exception.
E. An exception is thrown with no other output.
F. Compilation fails due to error(s) in the code.

24. Given:

```
 1. public class LaSelva extends Beach {
 2. LaSelva() { s = "LaSelva"; }
 3. public static void main(String[] args) { new LaSelva().go(); }
 4. void go() {
 5. Beach[] ba = { new Beach(), new LaSelva(), (Beach) new LaSelva() };
 6. for(Beach b: ba) System.out.print(b.getBeach().s + " ");
 7. }
 8. LaSelva getBeach() { return this; }
 9. }
10. class Beach {
```

```
11. String s;
12. Beach() { s = "Beach"; }
13. Beach getBeach() { return this; }
14. }
```

What is the result?

A. Beach LaSelva Beach
B. Beach LaSelva LaSelva
C. Beach LaSelva followed by an exception.
D. Compilation fails due to an error at line 5.
E. Compilation fails due to an error at line 6.
F. Compilation fails due to an error at line 8.
G. Compilation fails due to an error at line 13.

25. Given:

```
3. public class Stealth {
4. public static void main(String[] args) {
5. Integer i = 420;
6. Integer i2;
7. Integer i3;
8. i2 = i.intValue();
9. i3 = i.valueOf(420);
10. System.out.println((i == i2) + " " + (i == i3));
11. } }
```

What is the result?

A. true true
B. true false
C. false true
D. false false
E. Compilation fails.
F. An exception is thrown at runtime.

26. Given:

```
2. import java.io.*;
3. interface Risky {
4. String doStuff() throws Exception;
5. Risky doCrazy();
6. void doInsane();
7. }
8. class Bungee implements Risky {
9. public String doStuff() throws IOException {
10. throw new IOException();
11. }
12. public Bungee doCrazy() { return new Bungee(); }
13. public void doInsane() throws NullPointerException {
14. throw new NullPointerException();
15. } }
```

What is the result? (Choose all that apply.)

A. Compilation succeeds.

B. The Risky interface will not compile.
C. The Bungee.doStuff() method will not compile.
D. The Bungee.doCrazy() method will not compile.
E. The Bungee.doInsane() method will not compile.

27. Given that IllegalArgumentException extends RuntimeException, and given:

```
11. static String s = "";
12. public static void main(String[] args) {
13. try { doStuff(); }
14. catch (Exception ex) { s += "c1 "; }
15. System.out.println(s);
16. }
17. static void doStuff() throws RuntimeException {
18. try {
19. s += "t1 ";
20. throw new IllegalArgumentException();
21. }
22. catch (IllegalArgumentException ie) { s += "c2 "; }
23. throw new IllegalArgumentException();
24. }
```

What is the result?
A. c1 t1 c2
B. c2 t1 c1
C. t1 c1 c2
D. t1 c2 c1
E. Compilation fails.
F. An uncaught exception is thrown at runtime.

28. Given:

```
1. public class Networking {
2. public static void main(String[] args) {
3. List<Integer> i = new LinkedList<Integer>();
4. i.add(4); i.add(2); i.add(5);
5. int r = 1;
6. doStuff(r);
7. System.out.println(i.get(r));
8. }
9. static int doStuff(int x) {
10. return ++x;
11. } }
```

What is the result?
A. 2
B. 4
C. 5
D. Compilation fails.

E. An exception is thrown at runtime.

29. Given:

```
2. class Weed {
3. protected static String s = "";
4. final void grow() { s += "grow "; }
5. static final void growFast() { s += "fast "; }
6. }
7. public class Thistle extends Weed {
8. void grow() { s += "t-grow "; }
9. void growFast() { s+= "t-fast "; }
10. }
```

Which are the FEWEST change(s) required for this code to compile? (Choose all that apply.)
A. s must be marked public.
B. Thistle.grow() must be marked final.
C. Weed.grow() must NOT be marked final.
D. Weed.growFast() must NOT be marked final.
E. Weed.growFast() must NOT be marked static.
F. Thistle.growFast() must be removed from the class.

30. Given the following pseudo-code design for a new accounting system:

```
class Employee
 maintainEmployeeInfo()
 connectToRDBMS()
class Payroll
 setStateTaxCodes()
 findEmployeesByState()
class Utilities
 getNetworkPrinter()
```

Assuming the class and method names provide good definitions of their own functionalities, which are probably true? (Choose all that apply.)
A. These classes appear to have low cohesion.
B. These classes appear to have high cohesion.
C. These classes appear to have weak validation.
D. These classes appear to have strong validation.
E. These classes appear to have weak encapsulation.
F. These classes appear to have strong encapsulation.